4.4 Browser variations

Consider using a browser that is not installed in your system. You may resort to a Docker image very easily (see Docker browsers section).
Note that, in this case, the WebDriver will connect to a remote browser (no longer direct communication) and you should Docker installed in your system.

Lab 5 Behavior-driven development (Cucumber)

Context and key points

You may configure IntelliJ to offer extended support to run Cucumber.

Key Points

- The Cucumber framework enables the concept of "executable specifications": with Cucumber we use **examples** to specify what we want the software to do. The (feature) **scenarios are written before production code**.
- Cucumber executes features (test scenarios) written with the Gherkin language (readable by non-programmers too).
- The steps included in the feature description (scenario) must be mapped into Java test code by annotating test methods with matching "expressions". Expressions can be (traditional) regular expressions or the (new) Cucumber expressions.

Explore

- Cucumber school: guided exercises, video lessons,…
- BDD with Cucumber – video presentation.

5.1 Getting started

a) Get the "Java Skelton" (maven) project proposed by the Cucumber team.
Remember to make the necessary adjustments:
- change the Artifact group and Id
- review the compiler version in the POM (1.8 → 11 or 17)

Note some elements I the POM:

<dependencyManagement> with *-bom entries.	There are "bills of materials". Their use in the Dependency management section is useful to enforce the coherent use of the versions of related artifacts. Note that in the <dependencies> section, the related artifacts omit the version specification.
maven-compiler-plugin	Sometimes, to use more recent constructions, you may need to control the maven compiler version.

b) Note the project structure, identify the location of the features and the tests.

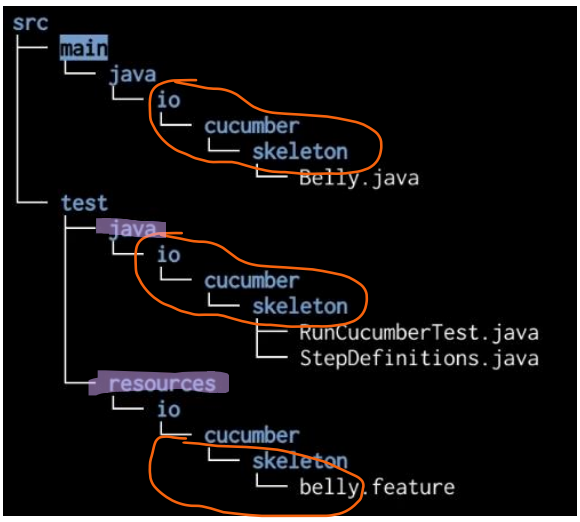
A feature will typical match a “test runner” class that will activate the test steps (in the same package).

The “test runner” class (that runs the test) should be annotated with:

@Suite

@IncludeEngines("cucumber")

@SelectClasspathResource("*my/package/structure*")



c) Run the tests.

The tests will not pass because they still are incomplete.

Note the “clues” in the output, giving suggestions to implement the missing steps (i.e., test methods). Use these suggestions to create the missing steps. Note that, for this purpose, the “Belly” class implementation can just log information.

d) Extend your project to include also the Calculator example discussed in “Cucumber in a Nutshell” section in B. Garcia’s book[4].

Note1: the example uses old-style integration with JUnit. You should adopt the strategy for the previous [“skeleton”] example. **No new imports or dependencies required**!

Note2: in this last example, the steps matching uses regular expressions. The best practice, however, is to use “cucumber expressions”. **Be sure to “upgrade”** the sample code. E.g.:

(old) regular expressions style →	(**better**) Cucumber expressions style
@When("^I add (\\d+) and (\\d+)$")	@When("I add {int} and {int}")

5.2 Books

To get into the “spirit” of BDD, partner with a colleague, and jointly write a couple of features to verify a book search user story. Consider a few search options (by author, by category, etc).

Take the approach discussed in this example, and write your own tests. Feel free to add different scenarios/features.

Notes:

- The article is based on old-style Cucumber constructions. Ignore the dependencies and annotations suggested in the text: use the “Skeleton” exercise as a reference for Cucumber dependencies and test annotations.
- Write the features before the test steps. Steps can be partially generated from features.
- Prefer the “cucumber expressions” (instead of regular expressions) to write the steps definitions.
- To handle dates, consider using a ParameterType configuration. This defines a new custom parameter type to use in the matching expressions . [→ partial snippet]. The dates in the feature description need also to match the date mask used (**aaaa-mm-dd**).
- To handle data tables in the feature description (as in the Salary manager example), consider using a DataTable mapping and access you data as a *list of maps* (use headings in the data).

[4] solution code available, if needed.

5.3 Web automation

Cucumber id often used with Selenium WebDriver to write expressive automation tests.

Consider the example available in B. Garcia's repository concerning the integration of Cucumber and Selenium [junit5-cucumber-selenium] and implement it.
Adapt from this example to create a test scenario related to the "BlazeDemo" application (recall Lab 4, exercise 2).